Appendix A

IETF IDN Working Group
Internet Draft
28 June 2000

Editors Zita Wenzel, James Seng
draft-ietf-idn-requirements-03.txt
Expires 28 November 2000

Requirements of Internationalized Domain Names

Abstract

This document describes the requirement for encoding international characters into DNS names and records. This document is guidance for developing protocols for internationalized domain names.

1. Introduction

At present, the encoding of Internet domain names is restricted to a subset of 7-bit ASCII (ISO/IEC 646). HTML, XML, IMAP, FTP, and many other text based items on the Internet have already been at least partially internationalized. It is important for domain names to be similarly internationalized or for an equivalent solution to be found. This document assumes that the most effective solution involves putting non-ASCII names inside some parts of the overall DNS system.

This document is being discussed on the "idn" mailing list. To join the list, send a message to <majordomo@ops.ietf.org> with the words "subscribe idn" in the body of the message. Archives of the mailing list can also be found at ftp://ops.ietf.org/pub/lists/idn*.

1.1 Definitions and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Characters mentioned in this document are identified by their position in the Unicode [UNICODE] character set. The notation U+12AB, for example, indicates the character at position 12AB (hexadecimal) in the Unicode character set. Note that the use of this notation is not an indication of a requirement to use Unicode.

Examples quoted in this document should be considered as a method to

further explain the meanings and principles adopted by the document. It is not a requirement for the protocol to satisfy the examples.

A character is a member of a set of elements used for organization, control, or representation of data.

A coded character is a character with its coded representation.

A coded character set ("CCS") is a set of unambiguous rules that establishes a character set and the relationship between the characters of the set and their coded representation.

A graphic character or glyph is a character, other than a control function, that has a visual representation normally handwritten, printed, or displayed.

A character encoding scheme or "CES" is a mapping from one or more coded character sets to a set of octets. Some CESs are associated with a single CCS; for example, UTF-8 [RFC2279] applies only to ISO 10646. Other CESs, such as ISO 2022, are associated with many CCSs.

A charset is a method of mapping a sequence of octets to a sequence of abstract characters. A charset is, in effect, a combination of one or more CCS with a CES. Charset names are registered by the IANA according to procedures documented in [RFC2278].

A language is a way that humans interact. In written form, a language is expressed in characters. The same set of characters can often be used in many languages, and many languages can be expressed using different scripts. A particular charset MAY have different glyphs (shapes) depending on the language being used.

1.2 Description of the Domain Name System

The Domain Name System is defined by [RFC1034] and [RFC1035], with clarifications, extensions and modifications given in [RFC1123], [RFC1996], [RFC2181], and others. Of special importance here is the security extensions described in [RFC2535] and companions.

Over the years, many different words have been used to describe the components of resource naming on the Internet (e.g., [URI], [URN]); to make certain that the set of terms used in this document are well-defined and non-ambiguous, the definitions are given here.

A master server for a zone holds the main copy of that zone. This copy is sometimes stored in a zone file. A slave server for a zone holds a complete copy of the records for that zone. Slave servers MAY be either authorized by the zone owner (secondary servers) or unauthorized (so-called "stealth secondaries"). Master and authorized slave servers are listed in the NS records for the zone, and are termed "authoritative" servers. In many contexts, outside this document the term "primary" is used interchangeably with "master" and "secondary" is used interchangeably with "slave".

A caching server holds temporary copies of DNS records; it uses records to answer queries about domain names. Further explanation of these terms can be found in [RFC1034] and [RFC1996].

DNS names can be represented in multiple forms, with different properties for internationalization. The most important ones are:

- Domain name: The binary representation of a name used internally in the DNS protocol. This consists of a series of components of 1-63

octets, with an overall length limited to 255 octets (including the length fields).

- Master file format domain name: This is a representation of the name as a sequence of characters in some character sets; the common convention (derived from [RFC1035] section 5.1) is to represent the octets of the name as ASCII characters where the octet is in the set corresponding to the ASCII values for [a-zA-Z0-9-], using an escape mechanism (\x or \NNN) where not, and separating the components of the name by the dot character (".").

The form specified for most protocols using the DNS is a limited form of the master file format domain name. This limited form is defined in [RFC1034] Section 3.5 and [RFC1123]. In most implementations of applications today, domain names in the Internet have been limited to the much more restricted forms used, e.g., in email. Those names are limited to the ASCII upper and lower-case characters (interpreted in a case-independent fashion), the digits, and the hyphen.

1.3 Definition of "hostname" and "Internationalized Domain Name"

In the DNS protocols, a name is referred to as a sequence of octets. However, when discussing requirements for internationalized domain names, what we are looking for are ways to represent characters that are meaningful for humans.

In this document, this is referred to as a "hostname". While this term has been used for many different purposes over the years, it is used here in the sense of "sequence of characters (not octets) representing a domain name conforming to the limited hostname syntax".

This document attempts to define the requirements for an "Internationalized Domain Name" (IDN). This is defined as a sequence of characters that can be used in the context of functions where a hostname is used today, but contains one or more characters that are outside the set of characters specified as legal characters for host names.

1.4 A multilayer model of the DNS function

The DNS can be seen as a multilayer function:

- The bottom layer is where the packets are passed across the Internet in a DNS query and a DNS response. At this level, what matters is the format and meaning of bits and octets in a DNS packet.

- Above that is the "DNS service", created by an infrastructure of DNS servers, NS records that point to those DNS servers, that is pointed to by the root servers (listed in the "root cache file" on each DNS server, often called "named.cache". It is at this level that the statement "the DNS has a single root" [RFC2826] makes sense, but still, what are being transferred are octets, not characters.

- Interfacing to the user is a service layer, often called "the resolver library", and often embedded in the operating system or system libraries of the client machines. It is at the top of this layer that the API calls commonly known as "gethostbyname" and "gethostbyaddress" reside. These calls are modified to support IPv6 [RFC2553]. A conceptually similar layer exists in authoritative DNS servers, comprising the parts that generate "meaningful" strings in DNS files. Due to the popularity of the "master file" format, this layer often exists only in the administrative routines of the service maintainers.

- The user of this layer (resolver library) is the application programs

that use the DNS, such as mailers, mail servers, Web clients, Web servers, Web caches, IRC clients, FTP clients, distributed file systems, distributed databases, and almost all other applications on TCP/IP.

Graphically, one can illustrate it like this:

```
+----------------+
| Application    |                            +----------------------+
+----------------+                            | (Base data)          |
      |  Application service interface        +----------------------+
      |  For ex. GethostbyXXXX interface                 |
+----------------+                                       | (no standard)
| Resolver       |                            +----------------------+
+----------------+                            | Auth DNS server      |
      |     <-----   DNS service interface    +----------------------+
      |                                       ----->     |
+------------------------------------------------------------------------+
|  DNS service                                                           |
|  +-----------------------+        +---------------------+              |
|  | Forwarding DNS server |        | Caching DNS server  |              |
|  +-----------------------+        +---------------------+              |
|                                                                        |
|                 +-------------------------+                            |
|                 | Parent-zone DNS servers |                            |
|                 +-------------------------+                            |
|                                                                        |
|                 +-------------------------+                            |
|                 | Root DNS servers        |                            |
|                 +-------------------------+                            |
|                                                                        |
+------------------------------------------------------------------------+
```

1.5 Service model of the DNS

The Domain Name Service is used for multiple purposes, each of which is characterized by what it puts into the system (the query) and what it expects as a result (the reply).

The most used ones in the current DNS are:

- Hostname-to-address service (A, AAAA, A6): Enter a hostname, and get back an IPv4 or IPv6 address.

- Hostname-to-Mail server service (MX): As above, but the expected return value is a hostname and a priority for SMTP servers.

- Address-to-hostname service (PTR): Enter an IPv4 or IPv6 address (in in-addr.arpa or ip6.int form respectively) and get back a hostname.

- Domain delegation service (NS). Enter a domain name and get back nameserver records (designated hosts who provides authoritive nameservice) for the domain.

New services are being defined, either as entirely new services (IPv6 to hostname mapping using binary labels) or as embellishments to other services (DNSSEC returning information about whether a given DNS service is performed securely or not).

These services exist, conceptually, at the Application/Resolver interface, NOT at the DNS-service interface. This document attempts to set requirements for an equivalent of the "used services" given above, where "hostname" is replaced by "Internationalized Domain Name". This doesn't preclude the fact that IDN should work with any kind of DNS

queries. IDN is a new service. Since existing protocols like SMTP or HTTP use the old service, it is a matter of great concern how the new and old services work together, and how other protocols can take advantage of the new service.

2. General Requirements

These requirements address two concerns: The service offered to the users (the application service), and the protocol extensions, if needed, added to support this service.

In the requirements, we attempt to use the term "service" whenever a requirement concerns the service, and "protocol" whenever a requirement is believed to constrain the possible implementation.

2.1 Compatibility and Interoperability

[1] The DNS is essential to the entire Internet. Therefore, the service MUST NOT damage present DNS protocol interoperability. It MUST make the minimum number of changes to existing protocols on all layers of the stack. It MUST continue to allow any system anywhere to resolve any internationalized domain name.

[2] The service MUST preserve the basic concept and facilities of domain names as described in [RFC1034]. It MUST maintain a single, global, universal, and consistent hierarchical namespace.

[2.5] The DNS service layer (the packet formats that go on the wire) MUST NOT limit the codepoints that can be used. This interface SHOULD NOT assign meaning to name strings; the application service layer, where "gethostbyname" et al reside, MAY constrain the name strings to be used in certain services. (conflict)

[3] The same name resolution request MUST generate the same response, regardless of the location or localization settings in the resolver, in the master server, and in any slave servers involved in the resolution process.

[4] The protocol SHOULD allow creation of caching servers that do not understand the charset in which a request or response is encoded. The caching server SHOULD perform correctly for IDN as well as for current domain names (without the authoritative bit) as the master server would have if presented with the same request.

[5] A caching server MUST NOT return data in response to a query that would not have been returned if the same query had been presented to an authoritative server. This applies fully for the cases when:

- The caching server does not know about IDN
- The caching server implements the whole specification
- The caching server implements a valid subset of the specification

[7] The service MAY modify the DNS protocol [RFC1035] and other related work undertaken by the [DNSEXT] WG. However, these changes SHOULD be as small as possible and any changes SHOULD be coordinated with the [DNSEXT] WG.

[8] The protocol supporting the service SHOULD be as simple as possible from the user's perspective. Ideally, users SHOULD NOT realize that IDN was added on to the existing DNS.

[10] The best solution is one that maintains maximum feasible compatibility with current DNS standards as long as it meets the other

requirements in this document.

2.2 Internationalization

[11] Internationalized characters MUST be allowed to be represented and used in DNS names and records. The protocol MUST specify what charset is used when resolving domain names and how characters are encoded in DNS records.

[12] This document RECOMMENDS Unicode only. If multiple charsets are allowed, each charset MUST be tagged and conform to [RFC2277].

[12.5] IDN MUST NOT return illegal code points in responses, SHOULD reject queries with illegal codepoints. (one request to add; one request to remove)

[13] CES(s) chosen SHOULD NOT encode ASCII characters differently depending on the other characters in the string. In other words, unless IDN names are identified and coded differently from ASCII-only ones, characters in the ASCII set SHOULD remain as specified in [US-ASCII] (one request to remove).

[14] The protocol SHOULD NOT invent a new CCS for the purpose of IDN only and SHOULD use existing CES. The charset(s) chosen SHOULD also be non-ambiguous.

[15] The protocol SHOULD NOT make any assumptions about the location in a domain name where internationalization might appear. In other words, it SHOULD NOT differentiate between any part of a domain name because this MAY impose restrictions on future internationalization efforts.

[16] The protocol also SHOULD NOT make any localized restrictions in the protocol. For example, an IDN implementation which only allows domain names to use a single local script would immediately restrict multinational organization.

[17] While there are a wide range of devices that use the DNS and a wide range of characteristics of international scripts and methods of domain name input and display, IDN is only concerned with the protocol. Therefore, there MUST be a single way of encoding an internationalized domain name within the DNS.

[18] The protocol SHOULD NOT place any restrictions on the application service layer. It SHOULD only specify changes in the DNS service layer and within the DNS itself.

2.4 Canonicalization

Matching rules are a complicated process for IDN. Canonicalization of characters MUST follow precise and predictable rules to ensure consistency. [CHARREQ] is RECOMMENDED as a guide on canonicalization.

The DNS has to match a host name in a request with a host name held in one or more zones. It also needs to sort names into order. It is expected that some sort of canonicalization algorithm will be used as the first step of this process. This section discusses some of the properties which will be REQUIRED of that algorithm.

[22] To achieve interoperability, canonicalization MUST be done at a single well-defined place in the DNS resolution process. The protocol MUST specify canonicalization; it MUST specify exactly where in the DNS that canonicalization happens and does not happen; it MUST specify how additions to ISO 10646 will affect the stability of the DNS and

the amount of work done on the root DNS servers.

[23] The canonicalization algorithm MAY specify operations for case, ligature, and punctuation folding.

[24] In order to retain backwards compatibility with the current DNS, the service MUST retain the case-insensitive comparison for [US-ASCII] as specified in [RFC1035]. For example, Latin capital letter A (U+0041) MUST match Latin small letter a (U+0061). [UTR21] describes some of the issues with case mapping. Case-insensitivity for non [US-ASCII] MUST be discussed in the protocol proposal.

[25] Case folding MUST be locale independent. For example, Latin capital letter I (U+0049) case folded to lower case in the Turkish context will become Latin small letter dotless i (U+0131). But in the English context, it will become Latin small letter i (U+0069).

[26] If other canonicalization is done, it MUST be done before the domain name is resolved. Further, the canonicalization MUST be easily upgradable as new languages and writing systems are added.

[27] Any conversion (case, ligature folding, punctuation folding, etc) from what the user enters into a client to what the client asks for resolution MUST be done identically on any request from any client.

[30] If the charset can be normalized, then it SHOULD be normalized before it is used in IDN. Normalization SHOULD follow [UTR15]. (conflict)

[31] The protocol SHOULD avoid inventing a new normalization form provided a technically sufficient one is available.

2.5 Operational Issues

[32] Zone files SHOULD remain easily editable.

[33] An IDN-capable resolver or server SHALL NOT generate more traffic than a non-IDN-capable resolver or server would when resolving an ASCII-only domain name. The amount of traffic generated when resolving an IDN SHALL be similar to that generated when resolving an ASCII-only name.

[34] The service SHOULD NOT add new centralized administration for the DNS. A domain administrator SHOULD be able to create internationalized names as easily as adding current domain names.

[35] Within a single zone, the zone manager MUST be able to define equivalence rules that suit the purpose of the zone, such as, but not limited to, and not necessarily, non-ASCII case folding, Unicode normalizations (if Unicode is chosen), Cyrillic/Greek/Latin folding, or traditional/simplified Chinese equivalence. Such defined equivalences MUST NOT remove equivalences that are assumed by (old or local-rule-ignorant) caches.

[36] The protocol MUST work with DNSSEC.

[37] The protocol MUST work for all features of DNS, IPv4, and IPv6.

4. Security Considerations

Any solution that meets the requirements in this document MUST NOT be less secure than the current DNS. Specifically, the mapping of internationalized host names to and from IP addresses MUST have the

same characteristics as the mapping of today's host names.

Specifying requirements for internationalized domain names does not itself raise any new security issues. However, any change to the DNS MAY affect the security of any protocol that relies on the DNS or on DNS names. A thorough evaluation of those protocols for security concerns will be needed when they are developed. In particular, IDNs MUST be compatible with DNSSEC and, if multiple charsets or representation forms are permitted, the implications of this name-spoof MUST be throughly understood.

5. References

[CHARREQ] "Requirements for string identity matching and String Indexing", http://www.w3.org/TR/WD-charreq, July 1998, World Wide Web Consortium.

[DNSEXT] "IETF DNS Extensions Working Group", namedroppers@internic.net, Olafur Gudmundson, Randy Bush.

[RFC1034] "Domain Names - Concepts and Facilities", rfc1034.txt, November 1987, P. Mockapetris.

[RFC1035] "Domain Names - Implementation and Specification", rfc1035.txt, November 1987, P. Mockapetris.

[RFC1123] "Requirements for Internet Hosts -- Application and Support", rfc1123.txt, October 1989, R. Braden.

[RFC1996] "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", rfc1996.txt, August 1996, P. Vixie.

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", rfc2119.txt, March 1997, S. Bradner.

[RFC2181] "Clarifications to the DNS Specification", rfc2181.txt, July 1997, R. Elz, R. Bush.

[RFC2277] "IETF Policy on Character Sets and Languages", rfc2277.txt, January 1998, H. Alvestrand.

[RFC2278] "IANA Charset Registration Procedures", rfc2278.txt, January 1998, N. Freed and J. Postel.

[RFC2279] "UTF-8, a transformation format of ISO 10646", rfc2279.txt, F. Yergeau, January 1998.

[RFC2535] "Domain Name System Security Extensions", rfc2535.txt, March 1999, D. Eastlake.

[RFC2553] "Basic Socket Interface Extensions for IPv6", rfc2553.txt, March 1999, R. Gilligan et al.

[RFC2825] "A Tangled Web: Issues of I18N, Domain Names, and the Other Internet protocols", rfc2825.txt, May 2000, L. Daigle et al.

[RFC2826] "IAB Technical Comment on the Unique DNS Root", rfc2826.txt, May 2000, Internet Architecture Board.

[IDNCOMP] "Comparison of Internationalized Domain Name Proposals", draft-ietf-idn-compare-00.txt, June 2000, P. Hoffman.

[UNICODE] The Unicode Consortium, "The Unicode Standard -- Version 3.0", ISBN 0-201-61633-5. Described at http://www.unicode.org/unicode/standard/versions/Unicode3.0.html

[US-ASCII] Coded Character Set -- 7-bit American Standard Code for Information Interchange, ANSI X3.4-1986.

[UTR15] "Unicode Normalization Forms", Unicode Technical Report #15, http://www.unicode.org/unicode/reports/tr15/, Nov 1999, M. Davis & M. Duerst, Unicode Consortium.

[UTR21] "Case Mappings", Unicode Technical Report #21, http://www.unicode.org/unicode/reports/tr21/, Dec 1999, M. Davis, Unicode Consortium. Approved status.

6. Editors' Contact

Zita Wenzel, Ph.D.
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA
90292 USA
Tel: +1 310 448 8462
Fax: +1 310 823 6714
zita@isi.edu

James Seng
8 Temesek Boulevand
#24-02 Suntec Tower 3
Singapore 038988
Tel: +65 248 6208
Fax: +65 248 6198
Email: jseng@pobox.org.sg

7. Acknowledgements

The editors gratefully acknowledge the contributions of:

Harald Tveit Alvestrand <Harald@Alvestrand.no>
Mark Andrews <Mark.Andrews@nominum.com>
RJ Atkinson <request not to have email>
Alan Barret <apb@cequrux.com>
Randy Bush <randy@psg.com>
Andrew Draper <ADRAPER@altera.com>
Martin Duerst <duerst@w3.org>
Patrik Faltstrom <paf@swip.net>
Ned Freed <ned.freed@innosoft.com>
Olafur Gudmundsson <ogud@tislabs.com>
Paul Hoffman <phoffman@imc.org>
Simon Josefsson <jas+idn@pdc.kth.se>
Karlsson Kent <keka@im.se>
John Klensin <klensin+idn@jck.com>
Tan Juay Kwang <tanjk@i-dns.net>
Dongman Lee <dlee@icu.ac.kr>
Bill Manning <bmanning@ISI.EDU>
Dan Oscarsson <Dan.Oscarsson@trab.se>
J. William Semich <bill@mail.nic.nu>
James Seng <jseng@pobox.org.sg>

Appendix B

Internet Draft
draft-ietf-idn-race-01.txt
August 31, 2000
Expires in six months

Paul Hoffman
IMC & VPNC

RACE: Row-based ASCII Compatible Encoding for IDN

Status of this memo

Abstract

This document describes a transformation method for representing non-ASCII characters in host name parts in a fashion that is completely compatible with the current DNS. It is a potential candidate for an ASCII-Compatible Encoding (ACE) for internationalized host names, as described in the comparison document from the IETF IDN Working Group. This method is based on the observation that many internationalized host name parts will have all their characters in one row of the ISO 10646 repertoire.

1. Introduction

There is a strong world-wide desire to use characters other than plain ASCII in host names. Host names have become the equivalent of business or product names for many services on the Internet, so there is a need to make them usable by people whose native scripts are not representable by ASCII. The requirements for internationalizing host names are described in the IDN WG's requirements document, [IDNReq].

The IDN WG's comparison document [IDNComp] describes three potential main architectures for IDN: arch-1 (just send binary), arch-2 (send binary or ACE), and arch-3 (just send ACE). RACE is an ACE, called Row-based ACE or RACE, that can be used with protocols that match arch-2 or arch-3. RACE specifies an ACE format as specified in ace-1 in [IDNComp]. Further, it specifies an identifying mechanism for ace-2 in [IDNComp], namely ace-2.1.1 (add hopefully-unique legal tag to the beginning of the name part).

Author's note: although earlier drafts of this document supported the ideas in arch-3, I no longer support that idea and instead only support arch-2. Of course, someone else might right an IDN proposal that matches

arch-3 and use RACE as the protocol.

In formal terms, RACE describes a character encoding scheme of the ISO 10646 [ISO10646] coded character set and the rules for using that scheme in the DNS. As such, it could also be called a "charset" as defined in [IDNReq].

The RACE protocol has the following features:

- There is exactly one way to convert internationalized host parts to and from RACE parts. Host name part uniqueness is preserved.

- Host parts that have no international characters are not changed.

- Names using RACE can include more internationalized characters than with other ACE protocols that have been suggested to date. Names in the Han, Yi, Hangul syllables, or Ethiopic scripts can have up to 17 characters, and names in most other scripts can have up to 35 characters. Further, a name that consist of characters from one non-Latin script but also contains some Latin characters such as digits or hyphens can have close to 33 characters.

It is important to note that the following sections contain many normative statements with "MUST" and "MUST NOT". Any implementation that does not follow these statements exactly is likely to cause damage to the Internet by creating non-unique representations of host names.

1.1 Terminology

The key words "MUST", "SHALL", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Hexadecimal values are shown preceded with an "0x". For example, "0xa1b5" indicates two octets, 0xa1 followed by 0xb5. Binary values are shown preceded with an "0b". For example, a nine-bit value might be shown as "0b101101111".

Examples in this document use the notation from the Unicode Standard [Unicode3] as well as the ISO 10646 names. For example, the letter "a" may be represented as either "U+0061" or "LATIN SMALL LETTER A".

RACE converts strings with internationalized characters into strings of US-ASCII that are acceptable as host name parts in current DNS host naming usage. The former are called "pre-converted" and the latter are called "post-converted".

1.2 IDN summary

Using the terminology in [IDNComp], RACE specifies an ACE format as specified in ace-1. Further, it specifies an identifying mechanism for ace-2, namely ace-2.1.1 (add hopefully-unique legal tag to the beginning of the name part).

RACE has the following length characteristics. In this list, "row" means a row from ISO 10646.

- If the characters in the input all come from the same row, up to 35 characters per name part are allowed.

- If the characters in the input come from two or more rows, neither of which is row 0, up to 17 characters per name part are allowed.

- If the characters in the input come from two rows, one of which is row 0, between 17 and 33 characters per name part are allowed.

1.3 Open issues

Is it OK in 2.3.2 to say "0 MAY be converted to O and that 1 MAY be converted to l"?

Do we want to leave the unused characters 0, 1, 8, and 9 "reserved" in Base32 instead of making them prohibited now? This allows creative expansion in the future.

2. Host Part Transformation

According to [STD13], host parts must be case-insensitive, start and end with a letter or digit, and contain only letters, digits, and the hyphen character ("-"). This, of course, excludes any internationalized characters, as well as many other characters in the ASCII character repertoire. Further, domain name parts must be 63 octets or shorter in length.

2.1 Name tagging

All post-converted name parts that contain internationalized characters begin with the string "bq--". (Of course, because host name parts are case-insensitive, this might also be represented as "Bq--" or "bQ--" or "BQ--".) The string "bq--" was chosen because it is extremely unlikely to exist in host parts before this specification was produced. As a historical note, in late August 2000, none of the second-level host name parts in any of the .com, .edu, .net, and .org top-level domains began with "bq--"; there are many tens of thousands of other strings of three characters followed by a hyphen that have this property and could be used instead. The string "bq--" will change to other strings with the same properties in future versions of this draft.

Note that a zone administrator might still choose to use "bq--" at the beginning of a host name part even if that part does not contain internationalized characters. Zone administrators SHOULD NOT create host part names that begin with "bq--" unless those names are post-converted names. Creating host part names that begin with "bq--" but that are not post-converted names may cause two distinct problems. Some display systems, after converting the post-converted name part back to an internationalized name part, might display the name parts in a possibly-confusing fashion to users. More seriously, some resolvers, after converting the post-converted name part back to an internationalized name part, might reject the host name if it contains illegal characters.

2.2 Converting an internationalized name to an ACE name part

To convert a string of internationalized characters into an ACE name part, the following steps MUST be preformed in the exact order of the subsections given here.

Note that if any checking for prohibited name parts (such as ones that are already legal DNS name parts), prohibited characters, case-folding, or canonicalization is to be done, it MUST be done before doing the conversion to an ACE name part. (Previous versions of this draft specified these steps.)

The input name string consists of characters from the ISO 10646 character set in big-endian UTF-16 encoding. This is the pre-converted

string.

Characters outside the first plane of characters (that is, outside the first 0xFFFF characters) MUST be represented using surrogates, as described in the UTF-16 description in ISO 10646.

2.2.1 Compress the pre-converted string

The entire pre-converted string MUST be compressed using the compression algorithm specified in section 2.4. The result of this step is the compressed string.

2.2.2 Check the length of the compressed string

The compressed string MUST be 36 octets or shorter. If the compressed string is 37 octets or longer, the conversion MUST stop with an error.

2.2.3 Encode the compressed string with Base32

The compressed string MUST be converted using the Base32 encoding described in section 2.5. The result of this step is the encoded string.

2.2.4 Prepend "bq--" to the encoded string and finish

Prepend the characters "bq--" to the encoded string. This is the host name part that can be used in DNS resolution.

2.3 Converting a host name part to an internationalized name

The input string for conversion is a valid host name part. Note that if any checking for prohibited name parts (such as ones that are already legal DNS name parts), prohibited characters, case-folding, or canonicalization is to be done, it MUST be done after doing the conversion from an ACE name part. (Previous versions of this draft specified these steps.)

2.3.1 Strip the "bq--"

The input string MUST begin with the characters "bq--". If it does not, the conversion MUST stop with an error. Otherwise, remove the characters "bq--" from the input string. The result of this step is the stripped string.

2.3.2 Decode the stripped string with Base32

The entire stripped string MUST be checked to see if it is valid Base32 output. The entire stripped string MUST be changed to all lower-case letters and digits. If any resulting characters are not in Table 1, the conversion MUST stop with an error; the input string is the post-converted string. Otherwise, the entire resulting string MUST be converted to a binary format using the Base32 decoding described in section 2.5. The result of this step is the decoded string.

2.3.3 Decompress the decoded string

The entire decoded string MUST be converted to ISO 10646 characters using the decompression algorithm described in section 2.4. The result of this is the internationalized string.

2.4 Compression algorithm

The basic method for compression is to reduce a full string that consists of characters all from a single row of the ISO 10646

repertoire, or all from a single row plus from row 0, to as few octets as possible. Any full string that has characters that come from two rows, neither of which are row 0, or three or more rows, has all the octets of the input string in the output string.

If the string comes from only one row, compression is to one octet per character in the string. If the string comes from only one row other than row 0, but also has characters only from row 0, compression is to one octet for the characters from the non-0 row and two octets for the characters from row 0. Otherwise, there is no compression and the output is a string that has two octets per input character.

The compressed string always has a one-octet header. If the string comes from only one row, the header octet is the upper octet of the characters. If the string comes from only one row other than row 0, but also has characters only from row 0, the header octet is the upper octet of the characters from the non-0 row. Otherwise, the header octet is 0xD8, which is the upper octet of a surrogate pair. Design note: It is impossible to have a legal stream of UTF-16 characters that has all the upper octets being 0xD8 because a character whose upper octet is 0xD8 must be followed by one whose upper octet is in the range 0xDC through 0xDF.

Although the two-octet mode limits the number of characters in a RACE name part to 17, this is still generally enough for almost all names in almost scripts. Also, this limit is close to the limits set by other encoding proposals.

Note that the compression and decompression rules MUST be followed exactly. This requirement prevents a single host name part from having two encodings. Thus, for any input to the algorithm, there is only one possible output. An implementation cannot chose to use one-octet mode or two-octet mode using anything other than the logic given in this section.

2.4.1 Compressing a string

The input string is in UTF-16 encoding with no byte order mark.

Design note: No checking is done on the input to this algorithm. It is assumed that all checking for valid ISO 10646 characters has already been done by a previous step in the conversion process.

Design note: In step 5, 0xFF was chosen as the escape character because it appears in the fewest number of scripts in ISO 10646, and therefore the "escaped escape" will be needed the least. 0x99 was chosen as the second octet for the "escaped escape" because the character U+0099 has no value, and is not even used as a control character in the C1 controls or in ISO 6429.

1) Read each pair of octets in the input stream, comparing the upper octet of each. If all of the upper octets (called U1) are the same, go to step 4.

2) Read each pair of octets in the input stream, comparing the upper octet of each. If all of the upper octets are either 0 or one single other value (called U1), go to step 4.

3) Output 0xD8, followed by the entire input stream. Finish.

4) Output U1.

5) If you are at the end of the input string, finish. Otherwise, read

the next octet, called U2, and the octet after that, called N1.

6) If U2 is equal to U1, and N1 is not equal to 0xFF, output N1, and go to step 5.

7) If U2 is equal to U1, and N1 is equal to 0xFF, output 0xFF followed by 0x99, and go to step 5.

8) Output 0xFF followed by N1. Go to step 5.

2.4.2 Decompressing a string

1) Read the first octet of the input string. Call the value of the first octet U1. If U1 is 0xD8, go to step 7.

2) If you are at the end of the input string, finish. Otherwise, read the next octet in the input string, called N1. If N1 is 0xFF, go to step 4.

3) Output U1 followed by N1. Go to step 2.

4) If you are at the end of the input string, stop with an error.

5) Read the next octet of the input string, called N1. If N1 is 0x99, output U1 followed by 0xFF, and go to step 2.

6) Output 0x00 followed by N1. Go to step 2.

7) Read the rest of the input stream and put it in the output stream. Finish.

2.4.3 Compression examples

For the input string of <U+012E><U+0110><U+014A>, all characters are in the same row, 0x01. Thus, the output is 0x012E104A.

For the input string of <U+012E><U+00D0><U+014A>, the characters are all in row 0x01 or row 0x00. Thus, the output is 0x012EFFD04A.

For the input string of <U+1290><U+12FF><U+120C>, the characters are all in row 0x12. Thus, the output is 0x1290FF990C.

For the input string of <U+012E><U+00D0><U+24C3>, the characters are from two rows other than 0x00. Thus, the output is 0xD8012E00D024C3.

2.5 Base32

In order to encode non-ASCII characters in DNS-compatible host name parts, they must be converted into legal characters. This is done with Base32 encoding, described here.

Table 1 shows the mapping between input bits and output characters in Base32. Design note: the digits used in Base32 are "2" through "7" instead of "0" through "6" in order to avoid digits "0" and "1". This helps reduce errors for users who are entering a Base32 stream and may misinterpret a "0" for an "O" or a "1" for an "l".

Table 1: Base32 conversion

bits	char	hex	bits	char	hex
00000	a	0x61	10000	q	0x71
00001	b	0x62	10001	r	0x72
00010	c	0x63	10010	s	0x73
00011	d	0x64	10011	t	0x74

00100	e	0x65	10100	u	0x75
00101	f	0x66	10101	v	0x76
00110	g	0x67	10110	w	0x77
00111	h	0x68	10111	x	0x78
01000	i	0x69	11000	y	0x79
01001	j	0x6a	11001	z	0x7a
01010	k	0x6b	11010	2	0x32
01011	l	0x6c	11011	3	0x33
01100	m	0x6d	11100	4	0x34
01101	n	0x6e	11101	5	0x35
01110	o	0x6f	11110	6	0x36
01111	p	0x70	11111	7	0x37

2.5.1 Encoding octets as Base32

The input is a stream of octets. However, the octets are then treated as a stream of bits.

Design note: The assumption that the input is a stream of octets (instead of a stream of bits) was made so that no padding was needed. If you are reusing this algorithm for a stream of bits, you must add a padding mechanism in order to differentiate different lengths of input.

1) Set the read pointer to the beginning of the input bit stream.

2) Look at the five bits after the read pointer. If there are not five bits, go to step 5.

3) Look up the value of the set of five bits in the bits column of Table 1, and output the character from the char column (whose hex value is in the hex column).

4) Move the read pointer five bits forward. If the read pointer is at the end of the input bit stream (that is, there are no more bits in the input), stop. Otherwise, go to step 2.

5) Pad the bits seen until there are five bits.

6) Look up the value of the set of five bits in the bits column of Table 1, and output the character from the char column (whose hex value is in the hex column).

2.5.2 Decoding Base32 as octets

The input is octets in network byte order. The input octets MUST be values from the second column in Table 1.

1) Set the read pointer to the beginning of the input octet stream.

2) Look up the character value of the octet in the char column (or hex value in hex column) of Table 1, and output the five bits from the bits column.

3) Move the read pointer one octet forward. If the read pointer is at the end of the input octet stream (that is, there are no more octets in the input), stop. Otherwise, go to step 2.

2.5.3 Base32 example

Assume you want to encode the value 0x3a270f93. The bit string is:

```
3   a    2   7    0   f    9   3
00111010 00100111 00001111 10010011
```

Broken into chunks of five bits, this is:

00111 01000 10011 10000 11111 00100 11

Padding is added to make the last chunk five bits:

00111 01000 10011 10000 11111 00100 11000

The output of encoding is:

```
00111 01000 10011 10000 11111 00100 11000
  h     i     t     q     7     e     y
```

or "hitq7ey".

3. Security Considerations

Much of the security of the Internet relies on the DNS. Thus, any change to the characteristics of the DNS can change the security of much of the Internet. Thus, RACE makes no changes to the DNS itself.

Host names are used by users to connect to Internet servers. The security of the Internet would be compromised if a user entering a single internationalized name could be connected to different servers based on different interpretations of the internationalized host name.

RACE is designed so that every internationalized host name part can be represented as one and only one DNS-compatible string. If there is any way to follow the steps in this document and get two or more different results, it is a severe and fatal error in the protocol.

4. References

[IDNComp] Paul Hoffman, "Comparison of Internationalized Domain Name Proposals", draft-ietf-idn-compare.

[IDNReq] James Seng, "Requirements of Internationalized Domain Names", draft-ietf-idn-requirement.

[ISO10646] ISO/IEC 10646-1:1993. International Standard -- Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane. Five amendments and a technical corrigendum have been published up to now. UTF-16 is described in Annex Q, published as Amendment 1. 17 other amendments are currently at various stages of standardization. [[[THIS REFERENCE NEEDS TO BE UPDATED AFTER DETERMINING ACCEPTABLE WORDING]]]

[RFC2119] Scott Bradner, "Key words for use in RFCs to Indicate Requirement Levels", March 1997, RFC 2119.

[STD13] Paul Mockapetris, "Domain names - implementation and specification", November 1987, STD 13 (RFC 1035).

[Unicode3] The Unicode Consortium, "The Unicode Standard -- Version 3.0", ISBN 0-201-61633-5. Described at <http://www.unicode.org/unicode/standard/versions/Unicode3.0.html>.

A. Acknowledgements

Mark Davis contributed many ideas to the initial draft of this document. Graham Klyne and Martin Duerst offered technical comments on the algorithms used. GIM Gyeongseog and Pongtorn Jentaweepornkul helped fix technical errors in early drafts.

Base32 is quite obviously inspired by the tried-and-true Base64 Content-Transfer-Encoding from MIME.

B. Changes from Versions -00 to -01 of this Draft

Throughout: Changed "ra--" to "bq--".

Throughout: Fixed minor typos.

1: Fixed the lengths allowed.

1.3: Fixed the lengths allowed.

2.1: Added note about changing the actual prefix in future versions of the draft.

2.4.1: Added first sentence. Changed steps that talked about characters to instead use "pair of octets". Fixed problem with the steps which caused bad output in some cases.

A: Added thanks to GIM Gyeongseog and Pongtorn Jentaweepornkul.

C. IANA Considerations

There are no IANA considerations in this document.

D. Author Contact Information

Paul Hoffman
Internet Mail Consortium and VPN Consortium
127 Segre Place
Santa Cruz, CA 95060 USA
paul.hoffman@imc.org and paul.hoffman@vpnc.org

United States Patent & Trademark Office

Office of Initial Patent Examination -- Scanning Division

Application deficiencies found during scanning:

☐ Page(s)___________ of ______________________________ were not present for scanning.

(Document title)

☐ Page(s)___________ of ______________________________ were not present for scanning.

(Document title)

☒ ***Scanned copy is best available.*** of drawings.